Assignment 1 – PROLOG part

Q3. Unification:

1. Error: functor Food(Y,soup) cannot start with an upper case letter.
2. Yes: Bread = soup.
3. Yes: For any case where Bread and Soup holds the same value.
4. No because in LHS, X should be salad, but on RHS, X should be milk.
5. No because the arity on LHS is not the same as the arity on RHS
6. Yes: X = healthyFood(bread) and Y = drink(milk)
8. Error: The list on the RHS has a misuse of “I” operator
9. Can be unified: X = l and Z =b
10. Yes: A = french(jean) and B = scottish(joe)
11. Yes: Y = drink(water) and X = healthyFood(bread)
12. Yes: H = a and T = [b, c]
13. No because the two lists are not of the same size.
14. Yes: Only if healthyFood(egg) unifies with healthyFood(bread). Then, Y=egg and Z = milk
15. Yes: X= jack, Y = cook(egg,oil), Time = Evening)
16. Yes: X = s(g) and Y = t(k)
17. Yes: Only if f(x,17,M) = f(x,x,M) = f(17,17,M), then Z = C, D = 17 or D = x, C = L\*y, E = 17
18. No: b is not a list, so it cannot be unified with [H|T]

Q4.

1. **? field(heat\_transfer,engineering).** : Ground
   1. The engine will try to match the query, one clause at a time starting from the top and going to the bottom until it is able to unify with the rule field(X,Y).
   2. Prolog will instantiate X to heat\_transfer and Y to engineering.
   3. Resolution will apply the substitution of variable in the rule, therefore creating a new rule: field(heat\_transfer,engineering):-course(heat\_transfer, Z), field(Z,engineering).
   4. The two new goals will be processed, and the engine will try to unify them for a certain value of Z. In this case, Prolog will find that the two goals are validated with Z = mechanical.
   5. The engine will therefore return **true**.
2. Non-Ground
3. **? field(computer, literature).** : Ground
   1. The engine will try to match the query, one clause at a time starting from the top and going to the bottom until it is able to unify with the rule field(X,Y).
   2. Prolog will instantiate X to computer and Y to literature.
   3. Resolution will apply the substitution of variable in the rule, therefore creating a new rule: field(computer,literature):-course(computer, Z), field(Z,literature).
   4. The two new goals will be processed and the engine will try to unify them for a certain value of Z. In this case, Prolog will fail to unify both goals.
   5. The engine will therefore return **false**.
4. Non-Ground
5. **? student(jeff).** : Ground
   1. The engine will try to match the query one clause at a time starting from the top and going to the bottom until it is able to unify with the rule student(X).
   2. Prolog will instantiate X to jeff.
   3. Resolution will apply the substitution of variable in the rule, therefore creating a new rule: student(john):- student (john,\_).
   4. The new goal will be processed, and the engine will try to unify it with another clause. In this case, Prolog will find that the goal unifies with the rule student(X, Y).
   5. Resolution will apply the substitution of variable in the rule, therefore creating a new rule: student(john, \_):- field (Z, \_), student(X, Z).
   6. The two new goals will be processed, and the engine will try to unify them for a certain value of Z. In this case, Prolog will find that the two goals are validated with Z = heat\_transfer. (The procedure for field(Z, \_) is described in 1.)
   7. The engine will therefore return **true**.
6. Ground
7. **? student(X, engineering).** : Non-Ground
   1. The engine will try to match the query one clause at a time starting from the top and going to the bottom until it is able to unify with the rule student(X,Y).
   2. Prolog will instantiate Y to engineering.
   3. Resolution will apply the substitution of variable in the rule, therefore creating a new rule: student(X, engineering):- :- field (Z, engineering), student(X, Z).
   4. The first goal will be processed, and the engine will try to unify it for any value of Z possible. In this case, Prolog will find that the goals are unified when Z = mechanical or Z = computer.
   5. The engine will then try to unify the second goal with Z = mechanical and find any values of X that validates the goal. In this case, no matching X will be found, so the goal will fail for this particular instance of Z.
   6. Now, the engine will try to unify the same goal with Z = computer and find any values of X that validates the goal. As in the previous step, no matching X will be found, so the goal will fail for all cases.
   7. Therefore, the engine will return **false.**
8. Non-Ground
9. **? field(\_, X).** : Non-Ground
   1. The engine will try to match the query, one clause at a time starting from the top and going to the bottom. Since the first argument can be matched to anything, Prolog will return any value of X for which the query unifies. In this case, any value.
   2. The engine will return: **X=engineering; X=engineering; X=art; X=social; X=business;**
   3. Prolog will then unify with the rule field(Y, Z), intantiating Y = \_ and Z = X.
   4. Resolution will apply the substitution of variable in the rule, therefore creating a new rule: field(\_,X):-course(\_, A), field(A,X).
   5. The two new goals will be processed and the engine will try to unify them for any value of X. The values that were returned previously are the same as the ones which validates the new goals with the exception of business.
   6. Therefore, the engine will return **X=engineering; X=engineering; X=art; X=social; false**.
10. Non-Ground
11. **? lab\_number(X, 15), field(X, Y).** : Non-Ground
    1. The engine will try to match the first clause of the query, one clause at a time starting from the top and going to the bottom of the data base until it is able to unify with the term lab\_number(mechanical, 15).
    2. Prolog will then instantiate X to mechanical and try to unify the second clause of the query. In this case, the engine will unify it with field(mechanical, engineer).
    3. Thus, the engine will return: **X = mechanical Y = engineer;**
    4. The engine will continue to unify the first clause of the query with one from the data base. It will be able to unify with the rule lab\_number(X,Z).
    5. Prolog will instantiate Z to 15.
    6. Resolution will apply the substitution of variable in the rule, therefore creating a new rule: lab\_number(X,15):- course(X,Y), lab\_number(Y,15)
    7. The two new goals will be processed, and the engine will try to unify them for a certain value of X and Y. In this case, X = heat\_transfer and Y = mechanical.
    8. Prolog will unify the second clause of the query with X = heat\_transfer. It will succeed to unify with field(heat\_transfer, engineering) via the rule field(X,Y) as shown in 1.
    9. Therefore, the engine will return: **X = heat\_transfer Y = engineer; false.**
12. Non-Ground
13. **? student(X), student(X,\_), !.** : Non-Ground
    1. The engine will try to match the first clause of the query, one clause at a time starting from the top and going to the bottom of the data base until it is able to unify with the rule student(X):-student(X,\_)
    2. The goal student(X,\_) will be able to unify with student(john, heat\_transfer). Therefore, the unification of the first clause of the query will be a success, so X will be instantiated with john.
    3. The second clause of the query student(john,\_) will now be verified, and will unify once again with student(john, heat\_transfer).
    4. The third statement of the query will be verified. Since it only consists of the cut operator, Prolog will end the unification process and will return: **X=john**.
14. Non-Ground